ООО «Цифровое проектирование»

Инструкция по установке программы для ЭВМ «Система управления стоимостью производства изделий Design To Cost (D2C)»

2025

АННОТАЦИЯ

Настоящий документ содержит информацию, необходимую для установки программы для ЭВМ «Система управления стоимостью производства изделий Design To Cost (D2C)» (далее также - D2C).

D2C

Оглавление

1. Общие сведения

Полное наименование программы для ЭВМ: Система управления стоимостью производства изделий Design To Cost (D2C)

Сокращённое наименование программы для ЭВМ: D2C

2. Подготовка серверов

2.1. Установка и настройка Сервера СУБД

Установить PostgreSQL 12 и выше. (Debian)

https://www.postgresql.org/download/

Настоящее решение поставляется в виде Docker контейнеров сервисов, необходимых для развёртывания в изолированной среде.

Для получения доступа к сервису необходима настройка DNS сервера.

Сервис состоит из четырех компонентов:

- NGINX веб-сервер, используется для предоставления доступа к сервису
- Frontend-сервис, используется для отображения графического интерфейса сервиса
- Backend-сервис, используется для выполнения необходимых вычислений, управления пользователями и для решения иных административных задач
- PostgreSQL СУБД, используется для хранения данных

Сервис поставляется в виде импортов образов Docker, необходимых для запуска и работы сервиса, архива volume содержащего БД, файла оркестратора контейнеров docker-compose.yaml и файла переменных окружения, необходимых для работы сервиса.

Настоящая инструкция написана по результатам развертывания на сервере с ОС Ubuntu Server 22.04 LTS.

2.2. Системные требования

1. Установленный Docker (Инструкция была воспроизведена на хосте с Docker version 27.0.3, build 7d4bcd8)

2. Установленный плагин Docker Compose (Инструкция была воспроизведена на хосте с Docker Compose version v2.28.1)
3. Доступ к DNS серверу, способному разрешать хосты в демонстрационном домене
4. Доступный для подключения клиентов порт 80, в случае развертывания без дополнительных прокси
5. 20 Гб свободного места на диске
6. 16 Гб оперативной памяти
7. 4 ядра CPU
8. У пользователя, от имени которого проводится развертывание, должна быть возможность повышать свои привилегии при помощи sudo, также желательно добавить его в группу docker, что бы избежать необходимости запуска контейнеров из под root

2.3. Порядок развертывания

1. Ниже подробное описание файлов, входящих в дистрибутив:
   a. Получение архивов образов, необходимых для запуска сервиса D2C.
      i. backend.tar(1,5Gb) – архив backend компоненты
      ii. busybox.tar(4.3M) – архив образа, используемого при импорте volume
      iii. db.tar(600Mb) – архив образа СУБД PostgreSQL
      iv. frontend.tar(1,8Gb) – архив frontend компоненты
      v. nginx.tar(183Mb) – архив nginx-прокси образа
   b. Получение архива БД сервиса.
      i. volume_backup.tar(115Mb) – архив дампа БД
   c. Получение файлов конфигурации развертывания и переменных окружения
      i. docker-compose.yaml
      ii. stages.env
      iii. nginx/config/nginx.conf
2. Настройка DNS сервера для предоставления доступа к сервису клиентам. В настоящей инструкции рассмотрено создание демонстрационной зоны на сервере имен BIND9. В случае использования иных серверов имен следует использовать этот пункт, как референсный.

Полный URL backend компонента сервиса используется при сборке кода frontend, следовательно в случае, если предложенный в инструкции URL (demo-D2C.local) не подходит, об этом необходимо сообщить разработчику, для перевыпуска им контейнера frontend и файла с переменными окружения stages.env

   a. Домен для демо-стенда: demo-D2C.local
   b. На сервере BIND9 создать зону:
      i. sudo nano /etc/bind/zones/db.demo-D2C.local

```
;
;BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     YOUR_DNS_SRV.DOMAIN.NAME. admin.DOMAIN.NAME. (
                        1           ; Serial
                   604800           ; Refresh
                    86400           ; Retry
                  2419200           ; Expire
                   604800 )         ; Negative Cache TTL
;
@       IN      NS      YOUR_DNS_SRV.DOMAIN.NAME.
@       IN      NS      YOUR_2ND_DNS_SRV.DOMAIN.NAME.
YOUR_DNS_SRV.DOMAIN.NAME.       IN      A       YOUR_DNS_IP
YOUR_2ND_DNS_SRV.DOMAIN.NAME.   IN      A       YOUR_2_DNS_IP
d2c-front.demo-D2C.local.       IN      A       SERVICE_IP
d2c-api.demo-D2C.local.         IN      A       SERVICE_IP
```

      ii. sudo nano /etc/bind/named.conf.local

```
zone "demo-D2C.local" {
        type master;
        file "/etc/bind/zones/db.demo-D2C.local";
};
```

      iii. sudo systemctl restart named
   c. Проверить доступность зоны как с хоста, на котором планируется развертывание так и с хостов клиентов
      i. nslookup d2c-front.demo-D2C.local (Ответ должен содержать IP адрес сервера, на котором будут доступны веб-интерфейсы сервиса)

3. На выбранном для развертывания сервере разрешить доступ по порту 80
4. В предлагаемом варианте развертывания файлы размещаются в директории /d2c, в случае необходимости директорию развертывания можно изменить, соответственно скорректировав настоящую инструкцию
5. Предполагается, что файл distributive.tar доставлен в директорию $HOME пользователя от имени которого проводится развертывание на целевом сервере.
6. Выполнить следующие действия. Действия представляют собой команды bash, пояснения к ним выделены при помощи символа #

```
# Создаем директорию, назначаем себя ее владельцем
sudo tar -xvf $HOME/distributive.tar -C /
```

```
sudo chown -R $USER:$USER /d2c
cd /d2c
# Начинаем процедуру загрузки образов
docker load < archives/busybox.tar
docker load < archives/backend.tar
docker load < archives/db.tar
docker load < archives/frontend.tar
docker load < archives/nginx.tar
# Экспортируем тэги Docker образов как переменные
export REST_IMAGE=d2c-demo-backend:v0.1.1
export DB_IMAGE=d2c-demo-db:v0.1.1
export FRONTEND_IMAGE=d2c-demo-frontend:v0.1.1
export NGINX_IMAGE=d2c-demo-nginx:v0.1.1

# Создаем volume и переносим в него данные из архива
# Запускаем только СУБД, в процессе развертывания контейнера будет выполнена инструкция по созданию volume
mkdir db_data
sudo chmod -R 777 db_data
docker compose run db ls -la /var/lib/postgresql/data
# Получаем имя созданного volume и сохраняем его как переменную
docker volume ls | grep db-data
➢ local     d2c_db-data(Имя volume)
export VOLUME_NAME=d2c_db-data
# Даем полные права на директорию с бекапом volume
sudo chmod -R 777 db_backup/
# Запускаем импорт volume
docker run --rm -v ${VOLUME_NAME}:/sample_data -v $(pwd)/db_backup:/backup busybox tar xvf /backup/volume_backup.tar -C /


# Запускаем контейнеры
docker-compose up -d

# Проверка состояния контейнеров
docker ps | grep d2c
```

7. Доступ к сервису
    a. Пользовательский интерфейс: d2c-front.demo-D2C.local
    b. Панель администратора: d2c-api.demo-D2C.local/admin

3. Приложения

3.1. docker-compose.yaml

```yaml
version: '3.8'

services:
 db:
  image: ${DB_IMAGE}
  restart: unless-stopped
  env_file:
   - './stages.env'
  volumes:
   - db-data:/var/lib/postgresql/data
  networks:
   - backend

 web:
  image: ${REST_IMAGE}
  restart: unless-stopped
  hostname: rest
  env_file:
   - './stages.env'
  environment:
   - RUN_REST=1
  depends_on:
   - db
  networks:
   backend:

 front:
  image: ${FRONTEND_IMAGE}
  restart: unless-stopped
  hostname: frontend
  env_file:
   - './stages.env'
  depends_on:
   - db
   - web
  networks:
   backend:

 proxy:
  image: ${NGINX_IMAGE}
  restart: unless-stopped
  hostname: proxy
  env_file:
   - './stages.env'
  ports:
   - "80:80"
  volumes:
```

```
    - ./nginx/config/nginx.conf:/etc/nginx/conf.d/default.conf:ro
  depends_on:
    - db
    - web
    - front
  networks:
    backend:

networks:
  backend:

volumes:
  db-data:
    driver: local
    driver_opts:
      type: none
      device: ./db_data
      o: bind
```

3.2. stages.env

```
SECRET_KEY="7e9uft1@53m%#!7^vuam4v*(9ez2aquvs7elnakuj=_m&h9^#5"
APPS_MODULE="d2c_mvp.wsgi"

DB_NAME=d2c_mvp
DB_HOST=db
DB_PORT=5432
DB_USER=postgres
DB_PASSWORD=postgres

POSTGRES_DB=${DB_NAME}
POSTGRES_USER=${DB_USER}
POSTGRES_PASSWORD=${DB_PASSWORD}
POSTGRES_PORT=${DB_PORT}
POSTGRES_HOST=${DB_HOST}

ND_PRODUCT=d2c
CI_COMMIT_REF_SLUG=master

API_URL=http://d2c-api.demo-D2C.local
PORT=80

S3_ENDPOINT=https://some-address.ru
S3_ACCESS_KEY=some-user
S3_SECRET_KEY=some-password
S3_BUCKET=some-bucket
S3_BACKUP_OBJ_PATH=some-archive.tar.gz
```

3.3. Dockerfile (backend)

```
FROM python:3.12.3

ARG PIP_INDEX_URL
```

```
ENV PIP_INDEX_URL=${PIP_INDEX_URL}

ENV HOME="/root"
ENV PATH="${HOME}/.local/bin:${PATH}"

RUN --mount=type=secret,id=netrc,dst=/root/.netrc pip install pipx
RUN --mount=type=secret,id=netrc,dst=/root/.netrc pipx install --force poetry==1.8.3
RUN poetry config virtualenvs.create false
RUN mkdir --parents /root/.local/pipx/venvs/poetry
RUN touch /root/.local/pipx/venvs/poetry/poetry_env
RUN poetry config virtualenvs.prefer-active-python true

COPY pyproject.toml poetry.lock /numdes/
WORKDIR /numdes

RUN --mount=type=secret,id=poetry,target=/root/.config/pypoetry/auth.toml \
    poetry install --verbose --no-root

COPY . .

RUN python --version
RUN python -m compileall -q .

ENTRYPOINT ["./docker/docker-entrypoint.sh"]
```

3.4. Dockerfile (frontend)

```
FROM node:21.2 as build

LABEL org.opencontainers.image.vendor="Numerical Design LLC"
LABEL org.opencontainers.image.description="D2C MVP Frontend Image"

ARG HOMEPAGE=/
ARG REACT_APP_ENVIRONMENT=development
ARG BASENAME
ARG APPS_HOME_URL=http://d2c-front.demo-D2C.local/

RUN echo "APPS_HOME_URL - $APPS_HOME_URL and BASENAME - $BASENAME"

WORKDIR /app

# Install dependencies
COPY package.json yarn.lock ./
RUN --mount=type=secret,id=npmrc,target=/root/.npmrc yarn install
ENV PATH=/app/node_modules/.bin:$PATH
ENV REACT_APP_ENVIRONMENT=${REACT_APP_ENVIRONMENT:-development}

# Build app
COPY .eslintrc.json .
COPY .prettierrc.json .
COPY tsconfig.json .

COPY src src/
```

```
COPY public public/
RUN ls -la
# Next we use sed to insert new line to package.json at line 2
# to declare that homepage\ will be a value of $HOMEPAGE variable
RUN sed -i "2i \"homepage\": \"${APPS_HOME_URL}\"," package.json
RUN yarn build

CMD ["yarn", "start"]
```

3.5. Dockerfile (PostgreSQL)

```
FROM postgis/postgis:15-
3.4@sha256:3efa7a5e4fc0484f52b372ed4a1949a7120742f8d98539802d1aa37b110a6842
MAINTAINER NumDes <info@numdes.com>
LABEL org.opencontainers.image.vendor="Numerical Design, LLC"
LABEL org.opencontainers.image.description="Docker image for PostGIS with restore from S3"
RUN apt-get update \
    && apt-get install --no-install-recommends -y \
    curl \
    ca-certificates \
    gettext-base \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/* \
    && curl --location "https://dl.min.io/client/mc/release/linux-amd64/mc" \
        --output /usr/local/bin/mcli  \
    && chmod +x /usr/local/bin/mcli
RUN mcli --version
ENV POSTGRES_DB="**None**" \
    POSTGRES_PORT=5432 \
    POSTGRES_USER=postgres \
    POSTGRES_PASSWORD="**None**" \
    S3_ACCESS_KEY="**None**" \
    S3_SECRET_KEY="**None**" \
    S3_ENDPOINT="**None**" \
    S3_BACKUP_OBJ_PATH="**None**" \
    LOCAL_DOWNLOAD_PATH="/tmp"
COPY docker-entrypoint-initdb.d/01-backup_restore_if_needed.sh docker-entrypoint-initdb.d/01-
backup_restore_if_needed.sh
HEALTHCHECK --interval=1s --timeout=3s --retries=15 \
    CMD pg_isready --username=$POSTGRES_USER --dbname=$POSTGRES_DB --port=$POSTGRES_PORT
|| exit 1
```